# **CS 410 Binary to C++ With Security Vulnerabilities Activity Template**

**Step 1:** Convert the binary file to assembly code.

**Step 2:** Explain the functionality of the blocks of assembly code.

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x0,-0x14(%rbp)  mov -0x14(%rbp),%eax | This section of the main function of the Assembly code declares the int main() function. This section also declares the variables option, a, and b that are all of type int. variable option is set to 0. |
| cmp $0x5,%eax  je 0xd02 <main+655>  lea 0x3a5(%rip),%rsi # 0xe49  lea 0x201575(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890  <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3a4(%rip),%rsi # 0xe5b  lea 0x201562(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x39c(%rip),%rsi # 0xe66  lea 0x20154f(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x399(%rip),%rsi # 0xe76  lea 0x20153c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x396(%rip),%rsi # 0xe86  lea 0x201529(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x346(%rip),%rsi # 0xe49  lea 0x201516(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x201623(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt> | This comparison is set to 5. This will be a while statement as it is broken down later in the assembly code. If the comparison is equal to 5, the program will exit  Load effective address of the string instruction. The call instruction will output the string  “ ----------------”  Load effective address of the string of rip at 0x39c bits to register rsi. This load will then be call with the instruction to output the string  “- 1)Add –“  Load effective address of rip at 0x399 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 2) Subtract- “  Load effective address of rip at 0x396 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 3) Multiply-“  Load effective address of the string instruction. The call instruction will output the string  “ ----------------”  Load effective address of rip at 0x346 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 4) Exit-“  This section loads the effective address of rbp at 14 bits to rax. Register rax is is then move to register rsi. This section is collecting user input and setting it to int option. |
| mov -0x14(%rbp),%eax  cmp $0x1,%eax  jne 0xbc3 <main+336>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x201604(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x2014c1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x327(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x30c(%rip),%rsi # 0xe96  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20141d(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30> | This section is the first part of conditional statements used to determine the user input and the process in which to execute the code. This section uses an if statement as comparison to 1.  If equal to one, the program will then collect user input for the first number. Then the user will input a second number.  After input is collected the system will output the statement “ a -b = answer”.  Once the outputs are displayed, the program will make an unconditional jump back to the beginning of the assembly code to gather user input for an option |
| mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 0xc62 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x201563(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x201420(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x286(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x26b(%rip),%rsi # 0xe96  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20137e(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30> | This section is the first part of conditional statements used to determine the user input and the process in which to execute the code. This section uses an if statement as comparison to 2.  If equal to two, the program will then collect user input for the first number. Then the user will input a second number.  After input is collected the system will output the statement “ a + b = answer”.  Once the outputs are displayed, the program will make an unconditional jump back to the beginning of the assembly code to gather user input for an option |
| mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 0xa91 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x2014c4(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x201381(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x1e7(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x1cc(%rip),%rsi # 0xe96  mov  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x2012de(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30>%rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi | This section is the first part of conditional statements used to determine the user input and the process in which to execute the code. This section uses an if statement as comparison to 3.  If equal to three, the program will then collect user input for the first number. Then the user will input a second number.  After input is collected the system will output the statement “ a / b = answer”.  Once the outputs are displayed, the program will make an unconditional jump back to the beginning of the assembly code to gather user input for an option |
| mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 0xd1b <main+680>  callq 0x8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq | This code block shows the comparison of the while statement. If the user input equals 5, the program will then make a jump and call instruction to exit the program. |
| DisplayMenu() Function  push %rbp  mov %rsp,%rbp  lea 0x400(%rip),%rsi # 0xe05  lea 0x201614(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3fe(%rip),%rsi # 0xe16  lea 0x201601(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3f5(%rip),%rsi # 0xe20  lea 0x2015ee(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3f1(%rip),%rsi # 0xe2f  lea 0x2015db(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3ed(%rip),%rsi # 0xe3e  lea 0x2015c8(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3a1(%rip),%rsi # 0xe05  lea 0x2015b5(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  nop  pop %rbp  retq | This is the Display Menu Function in the program. This is a separate function from the main function in the program. It identical to the first section of the main function to display a menu.  Load effective address of the string instruction. The call instruction will output the string  “ ----------------”  Load effective address of the string of rip at 0x39c bits to register rsi. This load will then be call with the instruction to output the string  “- 1)Add –“  Load effective address of rip at 0x399 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 2) Subtract- “  Load effective address of rip at 0x396 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 3) Multiply-“  Load effective address of rip at 0x346 bits to register rsi. This will load the string instruction that will be output with the call instruction to display “- 4) Exit-“  Load effective address of the string instruction. The call instruction will output the string  “ ----------------”  This section loads the effective address of rbp at 14 bits to rax. Register rax is is then move to register rsi. This section is collecting user input and setting it to int option. |

**Step 3:** Convert the assembly code to binary.

**Step 4:** Convert the assembly code to C++ code.

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax  movl $0x0,-0x14(%rbp)  mov -0x14(%rbp),%eax | int main() {  int option = 0;  int a;  int b; |
| cmp $0x5,%eax  je 0xd02 <main+655>  lea 0x3a5(%rip),%rsi # 0xe49  lea 0x201575(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890  <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3a4(%rip),%rsi # 0xe5b  lea 0x201562(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x39c(%rip),%rsi # 0xe66  lea 0x20154f(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x399(%rip),%rsi # 0xe76  lea 0x20153c(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x396(%rip),%rsi # 0xe86  lea 0x201529(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x346(%rip),%rsi # 0xe49  lea 0x201516(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x201623(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt> | while (option != 5) { // checking var option to output menu as long as option is not equal to 5  // This menu display is called in the displayMenu() function that is never called as well  cout << "----------------" << endl;  cout << "- 1)Add -" << endl; cout << "- 2)Subtract -" << endl;  cout << "- 3)Multiply -" << endl;  cout << "- 4)Exit -" << endl; // menu option 4 for exit. It is never initialized in program  cout << "----------------" << endl;  cin >> option; |
| mov -0x14(%rbp),%eax  cmp $0x1,%eax  jne 0xbc3 <main+336>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x201604(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x2014c1(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x327(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x30c(%rip),%rsi # 0xe96  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20141d(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30> | if (option == 1) {  cin >> a;  cin >> b;  cout << a << " - " << b << " = " << a - b << endl;  // wrong math action. should be addition. wrong symbol used. Should be "+" |
| mov -0x14(%rbp),%eax  cmp $0x2,%eax  jne 0xc62 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x201563(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x201420(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x286(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x26b(%rip),%rsi # 0xe96  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x20137e(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30> | else if (option == 2) {  cin >> a;  cin >> b;  cout << a << " - " << b << " = " << a + b << endl;  // wrong math action at end of output. should be subtraction |
| mov -0x14(%rbp),%eax  cmp $0x3,%eax  jne 0xa91 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x2014c4(%rip),%rdi # 0x202140 <\_ZSt3cin@@GLIBCXX\_3.4>  callq 0x870 <\_ZNSirsERi@plt>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  callq 0x870 <\_ZNSirsERi@plt>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x201381(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x1e7(%rip),%rsi # 0xe92  mov %rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  lea 0x1cc(%rip),%rsi # 0xe96  mov  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  callq 0x8d0 <\_ZNSolsEi@plt>  mov %rax,%rdx  mov 0x2012de(%rip),%rax # 0x201fd0  mov %rax,%rsi  mov %rdx,%rdi  callq 0x8a0 <\_ZNSolsEPFRSoS\_E@plt>  jmpq 0xa91 <main+30>%rax,%rdi  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi | else if (option == 3) {  cin >> a;  cin >> b;  cout << a << " - "<< b << " = "<< a/ b << endl;  // wrong math action. should be multiplication. wrong symbol used. Should be "\*" |
| mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 0xd1b <main+680>  callq 0x8b0 <\_\_stack\_chk\_fail@plt>  leaveq  retq | return 0; |
| DisplayMenu() Function  push %rbp  mov %rsp,%rbp  lea 0x400(%rip),%rsi # 0xe05  lea 0x201614(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3fe(%rip),%rsi # 0xe16  lea 0x201601(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3f5(%rip),%rsi # 0xe20  lea 0x2015ee(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3f1(%rip),%rsi # 0xe2f  lea 0x2015db(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3ed(%rip),%rsi # 0xe3e  lea 0x2015c8(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  lea 0x3a1(%rip),%rsi # 0xe05  lea 0x2015b5(%rip),%rdi # 0x202020 <\_ZSt4cout@@GLIBCXX\_3.4>  callq 0x890 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>  nop  pop %rbp  retq | void DisplayMenu(){  // function for display menu that is never called. Unused function in main  cout << "----------------" << endl;  cout << "- 1)Add -" << endl;  cout << "- 2)Subtract -" << endl;  cout << "- 3)Multiply -" << endl;  cout << "- 4)Exit -" << endl;  cout << "----------------" << endl;  } |